

Docket No. AUS920040031US1

**GRID MUTUAL AUTHORIZATION THROUGH PROXY CERTIFICATE  
GENERATION**

**BACKGROUND OF THE INVENTION**

**1. Technical Field:**

The present invention relates to data processing and, in particular, to grid computing. Still more particularly, the present invention provides a method, apparatus, and program for grid mutual authorization through proxy certificate generation.

**2. Description of Related Art:**

Grid computing is the sharing of central processing unit (CPU) resources across a network so that a plurality of machines function as one large supercomputer. Grid computing, also referred to as peer-to-peer computing or distributed computing, allows unused CPU capacity in any of the machines to be allocated to the total processing job required.

In a large enterprise, hundreds or thousands of desktop machines may be sitting idle at any given moment. A CPU may be considered idle even when a user is reading the screen and not interacting with the machine. Other computing resources may be idle at various times of the day. For example, a cluster of database servers, or a subset thereof, may be unused during non-business hours, for example during the middle of the night. Case in point, during peak business hours in New York City, most people are likely to be asleep in India. Therefore, at

Docket No. AUS920040031US1

any time of the day there may be unused processing cycles that could be put to use on large computational problems.

A given user may run a computationally intensive job using grid technology. For example, the user may have a biotechnology project, such as a human genome research project. A job may be any program that may run for two days or may simply ask the resource machine for the time of day. However, grid computing is typically used for very computationally intensive jobs. A user may send a job to a resource machine and go home for the night or the weekend. A resource machine will then process the job and return the results to the user.

The user may select a particular machine in a grid to run the job. The user may know of a specific machine that the user trusts. For example, the user may know of a machine with unused capacity that has updated security patches, a secure operating system, etc. Alternatively, a user may submit a job to a grid scheduler that selects a host (resource) to process the job. If a user sends a job to a grid scheduler, the user is presumed to trust the scheduler to send the job to the right host.

Of course, grid computing may bring up security problems. For instance, a resource machine may become overburdened. In this case, the primary resource machine may offload work to a secondary resource machine. The user that submitted the job may not necessarily trust the secondary resource or host. The IP of the job, the program being run, input data, output data, etc. are all available to the secondary untrusted host. This represents an unauthorized risk or cost to the user in

Docket No. AUS920040031US1

terms of security. However, once the job is submitted to a primary resource, the user has no control over which machine in the grid ultimately processes the job. Consequently, the secondary resource machine that processes the job may lack the desired, or even necessary, security levels. The machine may have an operating system that is vulnerable to virus attacks or spyware. Or the secondary resource may be malicious and steal the results or the Internet protocol (IP) address of the grid job.

**SUMMARY OF THE INVENTION**

The present invention recognizes the disadvantages of the prior art and provides a mechanism for mutual authorization of a secondary resource in a grid of resource computers. Grid security often uses certificates and proxy certificates to identify users. When a primary resource machine attempts to offload a grid computing job to a secondary resource, the primary resource machine sends a proxy certificate request to the user machine. Responsive to a proxy certificate request, the user machine performs an authorization job on the secondary resource machine. If the authorization with the secondary resource machine is successful, the user machine generates and returns a valid proxy certificate. The primary resource machine then performs mutual authentication with the secondary resource machine using the proxy certificate. However, if the authorization with the secondary resource machine is not successful, the user machine generates and returns an invalid proxy certificate. The primary resource then performs mutual authentication with the secondary resource machine; however, the authentication will fail due to the invalid proxy certificate. The primary resource will then select another secondary resource and repeat the process until a resource is found that passes the mutual authorization with the user machine.

**BRIEF DESCRIPTION OF THE DRAWINGS**

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

**Figure 1** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented;

**Figure 2** is a block diagram of a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

**Figure 3** is a block diagram of a data processing system in which the present invention may be implemented;

**Figure 4** illustrates a process for offloading a grid job from a primary resource to a secondary resource;

**Figure 5** illustrates a process for offloading a grid job from a primary resource to a secondary resource with grid mutual authorization through proxy certificate generation in accordance with a preferred embodiment of the present invention;

**Figure 6** is a block diagram depicting machines in a grid computing system in accordance with an exemplary embodiment of the present invention;

Docket No. AUS920040031US1

**Figure 7** is a flowchart illustrating the operation of a resource machine in a grid computing system in accordance with an exemplary embodiment of the present invention; and

**Figure 8** is a flowchart illustrating the operation of a user machine in a grid computing system in accordance with an exemplary embodiment of the present invention.

**DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT**

The present invention provides a method, apparatus and computer program product for mutual authorization of a secondary resource in a grid of resource computers. The data processing device may be a distributed data processing system in which multiple computing devices are utilized to perform various aspects of the present invention. Therefore, the following **Figures 1-3** are provided as exemplary diagrams of data processing environments in which the present invention may be implemented. It should be appreciated that **Figures 1-3** are only exemplary and are not intended to assert or imply any limitation with regard to the environments in which the present invention may be implemented. Many modifications to the depicted environments may be made without departing from the spirit and scope of the present invention.

With reference now to the figures, **Figure 1** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system **100** is a network of computers in which the present invention may be implemented. Network data processing system **100** contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within network data processing system **100**. Network **102** may include connections, such as wire, wireless communication links, or fiber optic cables.

Docket No. AUS920040031US1

In the depicted example, server **104** is connected to network **102** along with storage unit **106**. In addition, clients **108**, **110**, and **112** are connected to network **102**. These clients **108**, **110**, and **112** may be, for example, personal computers or network computers. In the depicted example, server **104** provides data, such as boot files, operating system images, and applications to clients **108-112**. Clients **108**, **110**, and **112** are clients to server **104**. Network data processing system **100** may include additional servers, clients, and other devices not shown.

More particularly, network data processing system **100** includes grid **120** that is a grid of computing systems. Grid computing is the sharing of central processing unit (CPU) resources across a network so that a plurality of machines function as one large supercomputer. Grid computing, also referred to as peer-to-peer computing or distributed computing, allows unused CPU capacity in any of the machines to be allocated to the total processing job required.

As shown in **Figure 1**, grid **120** may be made up of a combination of client and server computers. Any computer system with unused resources may be configured to accept grid jobs through network data processing system **100**. The grid computers within grid **120** may span the world and may be connected to network **102** through, for example without limitation, private networks, dial-up connections, dedicated Internet connections, gateway services, etc.

A given user may run a computationally intensive job using grid technology by submitting a job to one of the



Docket No. AUS920040031US1

resource machines in grid **120** using a workstation, such as client **108** in **Figure 1**. A resource machine will then process the job and return the results to the user. The user may select a particular machine in grid **120** to run the job. Alternatively, a user may submit a job to a grid scheduler that selects a host (resource) to process the job. A grid scheduler may be embodied on a machine in network data processing system **100** such as a server, server **104** for example. If a user sends a job to a grid scheduler, the user is presumed to trust the scheduler to send the job to the right host.

In the depicted example, network data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system **100** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 1** is intended as an example, and not as an architectural limitation for the present invention.

Referring to **Figure 2**, a block diagram of a data processing system that may be implemented as a server, such as server **104** in **Figure 1**, is depicted in accordance with a preferred embodiment of the present invention.

Docket No. AUS920040031US1

Data processing system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory controller/cache **208** and I/O bus bridge **210** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. A number of modems may be connected to PCI local bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to clients **108-112** in **Figure 1** may be provided through modem **218** and network adapter **220** connected to PCI local bus **216** through add-in connectors.

Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI local buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, data processing system **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 2** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in

Docket No. AUS920040031US1

place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 2** may be, for example, an IBM eServer pSeries system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

With reference now to **Figure 3**, a block diagram of a data processing system is shown in which the present invention may be implemented. Data processing system **300** is an example of a computer, such as client **108** in **Figure 1**, in which code or instructions implementing the processes of the present invention may be located. In the depicted example, data processing system **300** employs a hub architecture including a north bridge and memory controller hub (MCH) **308** and a south bridge and input/output (I/O) controller hub (ICH) **310**. Processor **302**, main memory **304**, and graphics processor **318** are connected to MCH **308**. Graphics processor **318** may be connected to the MCH through an accelerated graphics port (AGP), for example.

In the depicted example, local area network (LAN) adapter **312**, audio adapter **316**, keyboard and mouse adapter **320**, modem **322**, read only memory (ROM) **324**, hard disk drive (HDD) **326**, CD-ROM driver **330**, universal serial bus (USB) ports and other communications ports **332**, and PCI/PCIe devices **334** may be connected to ICH **310**. PCI/PCIe devices may include, for example, Ethernet adapters, add-in cards, PC cards for notebook computers,

Docket No. AUS920040031US1

etc. PCI uses a cardbus controller, while PCIe does not. ROM **324** may be, for example, a flash binary input/output system (BIOS). Hard disk drive **326** and CD-ROM drive **330** may use, for example, an integrated drive electronics (IDE) or serial advanced technology attachment (SATA) interface. A super I/O (SIO) device **336** may be connected to ICH **310**.

An operating system runs on processor **302** and is used to coordinate and provide control of various components within data processing system **300** in **Figure 3**. The operating system may be a commercially available operating system such as Windows XP, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provides calls to the operating system from Java programs or applications executing on data processing system **300**. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented programming system, and applications or programs are located on storage devices, such as hard disk drive **326**, and may be loaded into main memory **304** for execution by processor **302**. The processes of the present invention are performed by processor **302** using computer implemented instructions, which may be located in a memory such as, for example, main memory **304**, memory **324**, or in one or more peripheral devices **326** and **330**.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 3** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash memory, equivalent non-volatile

Docket No. AUS920040031US1

memory, or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 3**. Also, the processes of the present invention may be applied to a multiprocessor data processing system. Data processing system **300** also may be a tablet computer or laptop computer in addition to taking the form of a PDA.

A resource machine in a grid may become overburdened. In this case, a primary resource machine may offload work to a secondary resource machine. **Figure 4** illustrates a process for offloading a grid job from a primary resource to a secondary resource. The user performs authorization with the primary resource. The authorization is typically performed by the simple fact that the user sent the job to the host. If the user did not trust the host, then the user would not have sent the job to the host. That is, the user either initially trusts the primary resource or trusts a grid scheduler to assign the grid job to an appropriate primary resource. The resource authorizes the user by comparing the user's identification (ID) to an authorization list or gridmap file. Each resource in the grid keeps an authorization list or gridmap file that stores the ID and other attributes of each user that is authorized to run jobs on that resource. Often, an administrator for a grid pushes a list to all resources in the grid whenever a change is made to the list. After mutual authentication by exchanging the user's proxy certificate and the primary resource's host certificate, user **410** sends a grid job to primary resource **420**.

Docket No. AUS920040031US1

As illustrated in **Figure 4**, in operation **1**, primary resource **420** performs mutual authentication with the user machine **410**. Mutual authentication may be performed using digital certificates, such as X509 certificates, for example. User **410** may store a certificate, for example, `"/home/<user>/.globus/user_cert.pem."`

A typical digital certificate describes an end entity, such as a user, by distinguished name, for example (C=US, O=IBM, OU=AIX, CN=JohnUser). In this example, the digital certificate identifies a country (C), an organization (O), an organizational unit (OU), and a common name (CN). The digital certificate also contains the public key of the end entity and is signed by the private key of a trusted certificate authority (CA). Thus, when one receives the digital certificate, one may verify the signature of the trusted CA and trust that the CA properly vetted the identity of the user.

In operation **2**, primary resource **420** may determine that the grid job is to be offloaded to secondary resource **430**. The primary resource may offload a grid job if its processing resources are no longer underused. For example, the primary resource may receive workload for which the resource is intended, causing the primary resource to no longer have available CPU resources.

In operation **3**, the primary resource sends a certificate request to the user machine. User machine **410** then automatically generates a proxy certificate for the primary resource to use for authentication with the secondary resource. These processes are preferably automated, because these grid jobs or programs may run

Docket No. AUS920040031US1

for days. In other words, it is not practical for the user to be present to allow or deny these requests. User machine **410** may then store the proxy certificate, for example, `"/tmp/X509proxycert."`

A proxy certificate distinguished name is similar to a certificate with a string, such as `"/proxy,"` appended. For example, `(C=US,O=IBM,OU=AIX, CN=JohnUser/proxy)`. The proxy certificate is signed by the certificate, rather than by the CA. In other words, the CA signs the certificate and the certificate, in turn, signs the proxy certificate. When the proxy certificate is received by the remote grid resource, the chain of trust is as follows: 1) I trust the trusted CA and the trusted CA is signing the certificate; 2) Therefore, I trust the certificate; 3) I trust the certificate; therefore, I trust the proxy certificate signed by the certificate; and, 4) Therefore, I trust the proxy certificate.

In operation **4**, user machine **410** sends the proxy certificate to primary resource **420**. Thereafter primary resource **420** submits a job to secondary resource **430**. Mutual authentication then occurs in operation **5** and secondary resource **430** authorizes the user via proxy to run the job.

The advantage of the process shown in **Figure 4** is that a short lifetime is assigned to a proxy certificate. A typical lifetime may be, for example, twenty-four hours. The proxy certificate accompanies a grid job. Therefore, if the proxy certificate is compromised, it will expire within a short time, thus limiting the potential damage. Also, a grid job may move throughout

Docket No. AUS920040031US1

the grid without requiring the user to be attentive, because additional proxy certificates may be automatically generated as needed. This allows the user to submit a grid job with single sign-on no matter how the grid job is distributed within the grid. In other words, the user may simply submit a grid job and walk away.

The primary resource may select a secondary resource based on a set of criteria. In this respect, the primary resource may perform a form of authorization with the secondary resource. However, the user that submitted the job may not necessarily trust the secondary resource or host. Consequently, the secondary resource machine that processes the job may lack the desired, or even necessary, security levels. The machine may have an operating system that is vulnerable to virus attacks or spyware. Or the secondary resource may be malicious and steal the results or the Internet protocol (IP) address of the grid job.

**Figure 5** illustrates a process for offloading a grid job from a primary resource to a secondary resource with grid mutual authorization through proxy certificate generation in accordance with a preferred embodiment of the present invention. The premise behind grid mutual authorization through proxy certificate generation is to leverage or intercept the mutual authentication process to ascertain the following information:

1. The job is being offloaded or distributed beyond the primary resource;



Docket No. AUS920040031US1

2. The exact instance in time of this occurrence;  
and,

3. The derived IP address of the secondary resource.

With this information, the user machine running an automated process can discover the detailed information about the secondary resource, decide whether the secondary resource is authorized to run the job, and terminate or sabotage the secondary authentication process, if necessary, thus creating a mutual authorization mechanism.

User **510** sends a grid job to primary resource **520**. The user performs authorization with the primary resource. That is, the user either initially trusts the primary resource or trusts a grid scheduler to assign the grid job to an appropriate primary resource. In operation **1**, primary resource **520** performs mutual authentication with the user machine **510**. Mutual authentication may be performed using digital certificates, such as X509 certificates, for example.

In operation **2**, primary resource **520** may determine that the grid job is to be offloaded to secondary resource **530**. In operation **3**, the primary resource sends a certificate request to the user machine. Responsive to a certificate request from primary resource **520**, user machine **510** submits an authorization job in operation **3A** directly to the newly identified secondary resource **530**, independent of the current grid job being offloaded by primary resource **520**.

This authorization job runs on the secondary resource machine and identifies the security elements of

Docket No. AUS920040031US1

the secondary resource. For example, the authorization job may determine whether the secondary resource has the latest operating system and security updates, whether the secondary resource has a trusted operating system, whether there are any conflicting grid jobs also running on this resource, etc. The security information gathered by the authorization job may vary depending upon the implementation. For example, while a user may wish to list the jobs that a secondary resource is running on one implementation, a user may prefer a resource with this command disabled to avoid leaking critical and/or strategic information through covert channels.

The user machine may perform mutual authorization using more static decisions, such as having a black and/or white list, for example. The user machine may use rules, such as trust any resource trusted by the grid virtual organization, but do not trust a particular resource after business hours when monitoring is less intense. The authorization with the secondary resource is mutual, because the secondary resource authorizes the user by comparing a user ID to an authorization list, as discussed above.

If mutual authorization with the secondary resource is successful, user machine **510** generates a valid proxy certificate for the primary resource to use for authentication with the secondary resource. However, if the secondary resource fails the authorization process, user machine **510** generates an invalid proxy certificate.

In operation **4**, user machine **510** sends the proxy certificate to primary resource **520**. Thereafter primary

Docket No. AUS920040031US1

resource **520** submits a job to secondary resource **530**. Mutual authentication then occurs in operation **5** and secondary resource **530** authorizes the user via proxy to run the job. However, if primary resource **520** presents an invalid proxy certificate to secondary resource **530**, the secondary resource will not allow the user's grid job to be run. The primary resource will see this error and then search for another secondary resource to offload the job to, returning to operation **2**. Thus, the user machine performs mutual authorization with the secondary resource by forcing the secondary resource to either accept or reject the grid job through generation of a valid or invalid proxy certificate.

With reference now to **Figure 6**, a block diagram is shown depicting machines in a grid computing system in accordance with an exemplary embodiment of the present invention. User machine **610** includes authentication module **612**, certificate generation module **614**, and authorization module **612**. Primary resource machine **620** includes authentication module **622** and workload balancing module **624**. Secondary resource machine **630** includes authentication module **632** and workload balancing module **634**.

When primary resource machine **620** receives a grid job from user machine **610**, primary resource machine **620** performs authentication with the user machine using authentication module **622**. User machine **610** performs this mutual authentication with the primary resource machine through authentication module **612**. Primary resource machine **620** may also perform mutual

Docket No. AUS920040031US1

authentication with secondary resource **630** through authentication module **622** in the primary resource machine and authentication module **632** in the secondary resource machine.

Using workload balancing module **624**, primary resource machine **620** may decide to offload work to other resources. For example, primary resource machine **620** may offload a grid job that was received from user machine **610** to secondary resource machine **630**. Similarly, secondary resource machine **630** may then offload that same grid job to another resource machine (not shown) using workload balancing module **634**.

When primary resource machine **620** decides to offload a grid job that was received from user machine **610** to secondary resource machine **630**, primary resource machine **620** sends a certificate request to user machine **610**. In accordance with an exemplary embodiment of the present invention, user machine **610**, responsive to the certificate request from primary resource machine **620**, performs mutual authorization with secondary resource machine **630**.

The certificate request may identify the secondary resource machine, such as by IP address, for example. Alternatively, user machine **610** may submit a query grid job to primary resource machine **620** inquiring about the IP address of the secondary resource. User machine **610** then makes an authorization determination based on the secondary resource machine using authorization module **614**. User machine **610** may submit an authorization job to the newly identified secondary resource **630**. This

Docket No. AUS920040031US1

authorization job may run on the secondary resource machine and identify the security elements of the secondary resource. Authorization module **614** may also perform mutual authorization using more static decisions, such as having a black and/or white list, for example.

If mutual authorization with secondary resource machine **630** is successful, user machine **610** generates a valid proxy certificate using certificate generation module **616**. However, if secondary resource machine **630** fails the authorization process, user machine **610** generates an invalid proxy certificate using certificate generation module **616**.

User machine **610** sends the proxy certificate to primary resource machine **620**. Thereafter primary resource machine **620** submits a job to secondary resource machine **630**. Mutual authentication then occurs between authentication module **622** and authentication module **632** and secondary resource **630** authorizes the user via proxy to run the job. However, if primary resource machine **620** presents an invalid proxy certificate to secondary resource machine **630**, the secondary resource machine will not allow the user's grid job to be run. The primary resource machine will see this error and then search for another secondary resource to offload the job to using workload balancing module **624**.

**Figure 7** is a flowchart illustrating the operation of a resource machine in a grid computing system in accordance with an exemplary embodiment of the present invention. The process begins when request to process a grid job is received and performs mutual authentication

Docket No. AUS920040031US1

with the user machine (block **702**). Then, a determination is made as to whether the authentication is successful (block **704**). If the authentication is not successful, the process denies the grid job (block **706**) and ends.

If the authentication is successful in block **704**, the process authorizes the user (block **708**) and a determination is made as to whether authorization is successful (block **710**). If the authorization is not successful, the process denies the grid job (block **706**) and ends.

If the authorization is successful in block **710**, the process receives the grid job from the user machine (block **712**) and a determination is made as to whether to offload the grid job (block **714**). The determination in block **714** may be made at any time before the grid job is processed by the primary resource machine. If the grid job is not offloaded, the process runs the grid job (block **716**) and the process ends.

If the grid job is to be offloaded in block **714**, the process selects a secondary resource to which to send the grid job (block **718**). Then, the process sends a certificate request to the user machine (block **720**) and receives a proxy certificate from the user machine (block **722**). Thereafter, the process performs mutual authentication with the secondary resource using the proxy certificate (block **724**).

A determination is made as to whether authentication with the secondary resource is successful (block **726**). If authentication is not successful, the process returns to block **718** to select another secondary resource.

Docket No. AUS920040031US1

However, if authentication with the secondary resource is successful in block **726**, the process offloads the grid job to the secondary resource (block **728**) and the process ends.

**Figure 8** is a flowchart illustrating the operation of a user machine in a grid computing system in accordance with an exemplary embodiment of the present invention. The process begins when a grid job is to be submitted to a resource in a grid computing system. The process identifies a primary resource (block **802**) and performs mutual authentication with the primary resource (block **804**). A determination is made as to whether authentication is successful (block **806**). If authentication is not successful, the process returns to block **802** to identify another primary resource.

If authentication is successful in block **806**, the process performs authorization with the primary resource (block **808**). Then, a determination is made as to whether authorization with the primary resource is successful (block **810**). If authorization is not successful, the process returns to block **802** to identify another primary resource.

If authorization is successful in block **810**, the process sends the grid job to the primary resource (block **812**). A determination is made as to whether an exit condition exists (block **814**). An exit condition may exist, for example, when the user machine is shut down or the grid computing software is closed on the user machine. An exit condition may also exist when results

Docket No. AUS920040031US1

for the grid job are received. If an exit condition exists, the process ends.

If, however, an exit condition does not exist in block **814**, a determination is made as to whether a certificate request is received from the primary resource (block **816**). If a certificate request is not received, the process returns to block **814** to determine whether an exit condition exists.

If a certificate request is received from the primary resource in block **816**, the process performs authorization with the secondary resource (block **818**). Next, a determination is made as to whether authorization with the secondary resource is successful (block **820**). If authorization is successful in block **820**, the process generates a valid proxy certificate (block **822**); otherwise, the process generates an invalid proxy certificate (block **824**). After a valid or invalid proxy certificate is generated in one of blocks **822** and **824**, the process returns the proxy certificate to the primary resource. Thereafter, the process returns to block **814** to determine whether an exit condition exists.

Thus, the present invention solves the disadvantages of the prior art by providing a mechanism for grid mutual authorization through proxy certificate generation. Resource machines in a grid are able to authorize the user by comparing a user ID to an authorization list or gridmap file. The present invention allows the user machines to be configured to control the authorization of the distribution of its grid jobs without changing the grid software on the grid resources and still working



Docket No. AUS920040031US1

within the flow of the existing grid mutual authorization and one-way authorization. The mechanism of the present invention maintains a single sign-on paradigm while allowing the grid user to submit the job and walk away.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention,

Docket No. AUS920040031US1

the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.